ОДСЕК ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО

АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА 2

2021-2022

- трећи домаћи задатак -

Опште напомене:

1. Домаћи задатак 3 састоји се од два програмска проблема. Студенти проблеме решавају **самостално**, на програмском језику C++.
2. Реализовани програми треба да комуницира са корисником путем једноставног менија који приказује реализоване операције и омогућава сукцесивну примену операција у произвољном редоследу.
3. Унос података треба омогућити било путем читања са стандардног улаза, било путем читања из датотеке.
4. Решења треба да буду отпорна на грешке и треба да кориснику пруже јасно обавештење у случају детекције грешке.
5. Приликом оцењивања, биће узето у обзир рационално коришћење ресурса. **Примена рекурзије се неће признати као решење проблема које може освојити максималан број поена.**
6. За све недовољно јасне захтеве у задатку, студенти треба да усвоје разумну претпоставку у вези реализације програма. Приликом одбране, демонстраторе треба обавестити која претпоставка је усвојена (или које претпоставке су усвојене) и која су ограничења програма (на пример, максимална димензија низа и слично). Неоправдано увођење ограничавајуће претпоставке повлачи негативне поене.
7. Одбрана трећег домаћег задатка ће се обавити према распореду који ће накнадно бити објављен на сајту предмета. Пријава за одбрану биће омогућена преко Moodle система. Детаљније информације биће објављене на предметном сајту.
8. Предаја домаћих ће бити омогућена преко Moodle система. Детаљније информације ће бити благовремено објављене.
9. За решавање задатака који имају више комбинација користити следеће формуле. (**R** – редни број индекса, **G** – последње две цифре године уписа):

$$\mathbf{i = (R + G) \bmod 3}$$

$$\mathbf{j = (R + G) \bmod 2}$$

10. Предметни наставници задржавају право да изврше проверу сличности предатих домаћих задатака и коригују освојени број поена након одбране домаћих задатака, као и да пријаве теже случајеве повреде Правилника о дисциплинској одговорности студената Универзитета у Београду Дисциплинској комисији Факултета.

18.12.2021. године *Са предмета*

Задатак 1 – Хеш табела [70 поена]

Написати на језику C++ потребне класе за реализацију хеш табеле у коју се умећу објекти који садрже податке о студентима и њиховим пријављеним испитима. Подаци који се о студентима чувају су индекс студента, име и презиме и листа шифри предмета пријављених испита за тог студента. Подаци су индексирани (хеширани) на основу индекса студента (у формату **гггг**б**ббб** који се посматра као неозначени цео број, где је **гггг** пуна година уписа, а **бббб** број индекса) који је потребно користити као кључ за приступ табели. За разрешавање колизије се примењује техника отвореног адресирања. Примери CSV датотека које садрже податке о студентима дате су у прилогу. Потребно је реализовати класе **`HashTable`**, која представља хеш табелу и садржи операције за рад са истом и **`AddressFunction`**, која представља апстракцију адресне функције коју ће хеш табела користити за разрешавање колизије и задаје се објекту хеш табеле приликом њеног стварања. Такође, у зависности од добијене варијанте задатка потребно је реализовати и конкретну имплементацију те класе (описано у одговарајућем делу задатка).

[50 поена] Имплементација хеш табеле

Хеш табела која се користи за потребе овог задатка представљена је тако да један улаз у табелу представља један бакет величине *k*, тј. један улаз може садржати *k* кључева (задаје се приликом стварања). Хеш функција која се користи је $h = K \bmod 2^p$, тј. користе се најнижих *p* бита кључа. Функција је намерно одабрана да повећава вероватноћу колизија.

Приликом уметања кључа, додељена адресна функција **`AddressFunction`** се позива само ако је матична адреса заузета, ради разрешавања колизија. Класа **`HashTable`** треба да реализује следеће јавне методе:

- **[5 поена]** **`Info findKey(Key k)`** – проналази задати кључ и враћа показивач на одговарајући информациони садржај (невалидна вредност ако се кључ не налази у табели)
- **[10 поена]** **`bool insertKey(Key k, Info i)`** – умеће кључ и пратећи информациони садржај у табелу и враћа статус (***true*** за успешно уметање, ***false*** за неуспешно). Спречити уметање постојећег кључа.
- **[10 поена]** **`bool deleteKey(Key k)`** – брише кључ из табеле и враћа статус успеха (***true*** за успешно брисање, ***false*** за неуспешно). Брисање реализовати коришћењем технике полуслободних локација.
- **[5 поена]** **`void clear()`** – празни табелу (брише све кључеве)
- **[5 поена]** **`int keyCount()`** – враћа број уметнутих кључева
- **[5 поена]** **`int tableSize()`** – враћа величину табеле
- **[5 поена]** **`operator<<`** – испис садржаја табеле на стандардни излаз, у сваком реду по један улаз табеле. Празне улазе табеле означити са "**`EMPTY`**". Полуслободне улазе у табели означити са "**`DELETED`**".
- **[5 поена]** **`double fillRatio()`** – враћа степен попуњености табеле (реалан број између 0 и 1)

Исправна реализација хеш табеле подразумева да, поред наведених метода, постоје друге потребне методе (попут конструктора и деструктора). Студентима се препушта да у класу додају оне методе које сматрају потребним за успешну реализацију.

[15 поена] Имплементација апстрактне класе адресне функције и једне од метода отвореног адресирања

Апстрактна класа декларише јавну методу коју користи класа **`HashTable`** за одређивање наредне адресе приликом разрешавања колизије.

```
Address getAddress(Key k, Address a, Attempt i, Size n);
```

Параметри ове методе су:

`k` – кључ,
`a` – матична адреса,
`i` – редни број покушаја приступа,
`n` – величина хеш табеле.

Метода враћа нову, валидну адресу на којој треба потражити кључ (или локацију где га треба сместити). Изведене класе треба да конкретизују начин одређивања следеће адресе.

У зависности од редног броја проблема који се решава **i**, реализовати следећу методу за решавање колизија:

0. Линеарно претраживање са раздвојеном секвенцом (класа **`SplitSequenceLinearHashing`**):
   Класа за линеарно адресирање са раздвојеном секвенцом се параметризује кораком **`s1`** и кораком **`s2`**. Када се за дати кључ **`k2`** утврди да је његова матична адреса већ заузета неким другим кључем **`k1`** онда се ова два кључа упореде. Ако је **`k2 < k1`** за корак у табели се узме константа **`s1`**, а ако је **`k2 > k1`** узме се константа **`s2`** различита од **`s1`**. Поступак се понавља у сваком кораку хеширања.

   $$\texttt{return\_address} = a + s1 \cdot i$$
   или
   $$\texttt{return\_address} = a + s2 \cdot i$$

1. Квадратно претраживање (класа **`QuadriaticHashing`**):
   Класа за квадратно адресирање се параметризује коефицијентом **`c`**, а у **`i`** –том покушају враћа вредност према следећој формули:

   $$\texttt{return\_address} = a + c \cdot i^2$$

2. Двоструко хеширање (класа **`DoubleHashing`**):
   Класа за двоструко хеширање се параметризује подацима **`p`** и **`q`** и враћа следећу вредност:

   $$\texttt{return\_address} = a + i \cdot (q + (k \bmod p))$$

[5 поена] Главни програм

Реализовати главни програм са једноставним интерактивним менијем који кориснику омогућава рад са јавним методама хеш табеле. Ажурирање информационог садржаја студента треба реализовати кроз две опције – додавање испита у листу пријављених и брисање испита из листе пријављених. Такође, главни програм треба да омогући учитавање објеката који се хеширају са стандардног улаза или задате датотеке.

Уз поставку задатка доступне су датотеке које садрже податке о објектима (студенти) које треба хеширати и која се може користити за тестирање решења.

Задатак 2 - Имплементација флексибилних техника спољашњег хеширања [30 поена]

У случајевима да се број записа у табели динамички мења, може се десити да се перформансе погоршавају или да је простор неефикасно искоришћен. Тада је пожељно да се поступак хеширања прилагођава динамичкој промени броја кључева. Потребно је модификовати хеш табелу из првог задатка тако да се користи једна од флексибилних техника спољашњег хеширања.

У зависности од редног броја проблема који се решава **j**, реализовати следећу технику

0. Динамичко хеширање (задаје се b приликом стварања табеле, број бакета је $m=2^b$)
1. Проширљиво хеширање (задаје се почетна дубина b приликом стварања табеле)

Приликом задавања параметара за наведене технике, обратити пажњу на величину табеле. Потребно је да параметар p задат за хеш функцију буде довољно већи од параметра b, како би се ове технике могле успешно применити.